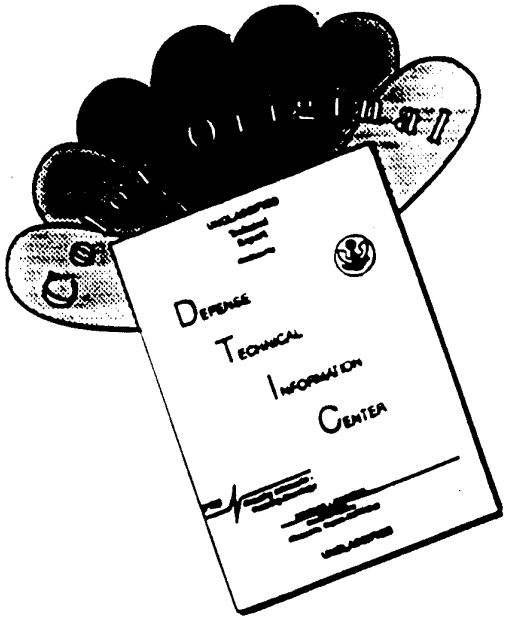
DSTO

Colour Graphics for Hydrocode Simulations

A. Doyle, D.A. Jones and G. Kemister

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

Colour Graphics for Hydrocode Simulations

A. Doyle, D.A. Jones and G. Kemister

Weapons Systems Division
Aeronautical and Maritime Research Laboratory

DSTO-GD-0059

ABSTRACT

This report describes the production of colour graphical output for the two-dimensional finite difference reactive hydrocodes currently used by Weapons Systems Division. The programs use the NCAR graphics package and show how the careful choice of appropriate colour schemes can significantly enhance the usefulness of the output from the finite difference codes. Applications to recent work in WSD are described, including simulation of the initiation of detonation in Composition B by bullet impact, and the reignition of gaseous detonation following an abrupt area change. The report also discusses several ways in which a sequence of colour plots can be animated and then transferred to videotape, and a videotape sequence of the time dependent cellular structure of a gaseous detonation front was made to illustrate the implementation of the recommended procedure. Now that the capabilities of the software have been fully demonstrated these colour graphical and video techniques will be used on Service Sponsored Tasks to more effectively communicate the results and outcomes to the sponsors.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001

Telephone: (03) 9626 8111
Fax: (03) 9626 8999

AR No. 009-358
January 1996

APPROVED FOR PUBLIC RELEASE

Colour Graphics for Hydrocode Simulations

Executive Summary

The production of colour graphical output for the two-dimensional finite difference reactive hydrocodes currently used by Weapons Systems Division (WSD) is described. These programs model airblast around structures, detonation initiation and failure in selected explosives, and explosive/material interaction effects. The previous method of analysing the output from the calculations was to produce monochrome contour plots using the NCAR plotting package. This was an effective method of viewing the data for staff familiar with the details of the work, but the monochrome plots used only a small fraction of the capabilities of the NCAR package, and were not acceptable for presentation purposes.

The graphics programs described here illustrate the use of NCAR's colour plotting capability, and show how the careful choice of appropriate colour schemes can significantly enhance the usefulness of the simulated data. The colour plots provide detailed information about the regions of interest, and enable faster and better understanding of the underlying physical processes. Applications to recent work in WSD are described which illustrate the usefulness of these techniques. These include simulating the initiation of detonation in Composition B by bullet impact, and the reignition of gaseous detonation following an abrupt area change.

The report also describes several ways in which a sequence of colour plots can be animated, and then transferred to videotape. The ability to view the time development of the data in a realistic time frame can play a significant role in the detailed understanding of the underlying transient processes. As an illustration, a videotape sequence of the time dependent cellular structure of a gaseous detonation front was made. This resulted in a much more accurate understanding of the mechanism responsible for the reignition of detonation following an abrupt area change.

Now that the capabilities of the software have been fully demonstrated these colour graphical and video techniques will be used on Service Sponsored Tasks to more effectively communicate the results and outcomes to the sponsors.

Contents

1. INTRODUCTION 1

2. NCAR ROUTINES 2
2.1 Contouring Routines 3
2.1.1 Contour Setting 3
2.1.2 A basic contouring routine 3
2.1.3 Area map contours 4
2.2 Area Filling Routines 5
2.3 Cell Arrays 7
2.4 Colour Maps 9
2.5 Label Bars 11
2.6 Aspect Ratio 11
2.7 Contouring Selected Regions 13
2.8 Printing 13

3. EXAMPLES 14
3.1 Bullet Impact 14
3.2 Gaseous Detonation 15
3.3 Triple Point Trajectories 21

4. VIDEO 24
4.1 Equipment - Scan Converter/Video Card 24
4.2 X Vision 24
4.3 Writing Frame by Frame to Video 25
4.4 Video for Windows 25
4.5 Video Output 28

5. LIMITATIONS AND EQUIPMENT REQUIREMENTS 29
5.1 Memory 29
5.2 Disk Space 29
5.3 PC Resolution and XVision 30
5.4 AVI Playback Rate 30
5.5 Writing to Video 30
5.6 Printing 31

6. CONCLUSIONS AND RECOMMENDATIONS 31
6.1 Discussion 31
6.2 Equipment Purchases 32

7. ACKNOWLEDGEMENTS 33

8. REFERENCES.....33

9. APPENDICES.....35
9.1 Bullet Impact Program.....35
9.2 Gaseous Detonation Program -Colour Contouring Method.....38
9.3 Gaseous Detonation Program - Cell Array Method.....44
9.4 Triple Point Trajectories Program.....49

1. Introduction

Weapons Systems Division of AMRL uses several specialised finite difference hydrocodes to model airblast around structures, the initiation of detonation in gaseous and condensed phase explosives, and underwater explosives effects. These codes provide a greater understanding of the science involved in the initiation and detonation of explosives, and allow predictions of blast overpressures to be made without recourse to expensive field trials.

The physical equations determining the flow field are solved on a finite difference grid. In all the examples described in this report the grids are two-dimensional, and the calculations are advanced over the grid in discrete time steps. Typically a large number of time steps (of the order of several thousand) are needed to provide a clear picture of the evolution of the flow. Output from the codes is in the form of a matrix of floating point numbers, with each number in the matrix representing the value of a chosen variable (eg. pressure or temperature) at the corresponding grid location, and at a given point in time. The large amount of data produced means that without appropriate visualisation techniques it is very difficult to quickly comprehend important features of the flow field. A typical 2D grid size is 640 x 640, which means over 409,600 grid locations, each with its associated numerical value. The codes are typically run for several thousand time steps, with results being dumped to file at selected intervals (for example, ranging from every 5 to 200 time steps). The dump file generated from a single run can be quite large, often containing several hundred megabytes of data.

A previous method of examining the data was to use the NCAR plotting package to produce a monochrome contour plot. This was an effective method of viewing the data for staff familiar with the details of the work, but the monochrome plots used only a small fraction of the capabilities of the NCAR package, and were not acceptable for presentation purposes. NCAR also has the ability to produce colour contour plots, and this report describes the application of the NCAR colour graphics software to the analysis of the output data from the AMRL hydrocodes. This enables the data to be more quickly analysed, enhances the presentation quality of the results, and in some cases can increase our understanding of the structure of the flow.

NCAR is a commercial plotting package which runs on a variety of operating systems. NCAR is installed on the WSD HP Workstations running under the UNIX operating system. All the WSD programs have been written using the FORTRAN programming language, and invoke various procedure calls to NCAR routines. The HP FORTRAN 9000 compiler is used to compile the FORTRAN code, and to link in the NCAR libraries.

A number of routines were written to read the dump files from the AMRL hydrocodes and display the data using the NCAR colour graphics capability. Section 2 contains a description of the relevant NCAR calls, while Section 3 contains examples of the colour output from selected applications of the codes. These include the numerical simulation

output from selected applications of the codes. These include the numerical simulation of bullet impact experiments, gaseous detonation ignition and failure, and the motion of triple-point trajectories. Complete listings of the plotting routines are contained in the appendices.

The output from the codes is time dependent, and in many cases the flow contains features which have a short lifetime during the simulation period, but which may have a significant effect on the final state of the system. In this case, it is highly desirable to animate the sequence of contour plots from successive time dumps, and then transfer the animation to video tape for future viewing. Two animation sequences were produced to illustrate this procedure, one for the bullet impact simulations, and one for the gaseous detonation calculations. Section 4 contains a detailed description of the method used to animate the time frames, and the method used to transfer the animation to video tape. Section 5 discusses the software and hardware requirements necessary for video animations of this type, as well as some of the limitations experienced during the course of the work. Finally, in Section 6, some recommendations are made for future work in this area.

2. NCAR Routines

NCAR is a graphics package that is based on the Graphics Kernel Standard (GKS). Before the NCAR routines can be called, GKS must be opened (*call opngks*), the data files must be opened, and the data read into an array in memory. Each time a picture is desired, the routine *frame* must be called. This stores the current state of the picture into a graphics metafile (with a default filename of *gmeta*) and resets the picture stored in memory. An example of this type of code is shown below:

```
      call opngks
      do 1, i=1,ncx
            read (10,*) (prs(i,j), j=1,ncy)
1     continue
      call cpcnrc(prs,ncx,ncx,ncy,0.0,30.0,2.0,1,-1,0)
      call frame
      call clsgks
```

NCAR has the ability to produce plots in several different ways. The simplest way of examining the data is to produce black and white contour plots of the arrays. This was the method previously used in examining the hydrocode output, and was also used in the current programs to overlay an outline of each of the different materials present on the grid when colour plots were produced for the multimaterial code.

The colour plots can be produced in one of two ways. The first method is to contour the data and then to fill the regions between the contour lines with colour. This method uses different contouring calls than the monochrome method, and relies on the use of an area map. The second method uses a cell array, rather than actually contouring the data. This method involves setting up a two dimensional array (the cell array), which stores the colour representing the value of each cell in the finite difference grid.

2.1 Contouring Routines

2.1.1 Contour Setting

Before calling a contouring routine, NCAR allows the user the option of automatic selection of the number and the levels of the contours from a default setting or for the manual setting of these numbers through the following routines. The user has the ability to manually set the number of contour levels by setting the parameter *NCL* in a call to the subroutine *cpseti*:

```
call cpseti('NCL - Number of Contour Levels', number_of_contours)
```

The user also has the ability to manually set the level of each contour. This is achieved by setting the parameter *CLV* to the desired contour level value for each contour. The contour number being set corresponds to the current value of *PAI* (the Parameter Array Index). The contour values are set in a DO loop, which calls the subroutine *cpsetr* the required number of times. The values that are being assigned to the contour levels can be stored in an array. This procedure is illustrated below:

```
      do 300, i=1, numcont
            call cpseti('PAI - Parameter Array Index', i)
            call cpsetr('CLV - Contour Level Values', contlevel(i))
300   Continue
```

2.1.2 A basic contouring routine

The routine *cpcnrc* is the basic contouring routine to be used when only black and white contours are required (for example when overlaying the material interface).

```
call cpcnrc (data, sizex, ncx, ncy, min, max, inc, nset, labs, dash)
```

A specific example of its use occurs in the bullet impact model when the array containing the volume fraction of the bullet (vfbull) is contoured.

```
C     Draw bullet outline
      call cpcnrc(vfbull,mcx,ncx,ncy,0.0,1.0,0.5,1,-1,0)
```

A detailed description of the parameter values to be used when calling *cpcnrc* is contained in Table 1.

Table 1: CPCNRC

Parameter	Description
data	Array containing the data to be contoured
sizex	First dimension of the data array (Not all of the data needs to be contoured, hence ncx can be smaller than sizex)
ncx	Size of dimension of data in the x direction
ncy	Size of dimension of data in the y direction
min	The desired lowest contour level
max	The desired highest contour level
inc	The desired contour interval. If inc = 0 then the interval will be chosen to give at least 16 contour levels. If inc < 0 then there will be -inc levels.
nset	Determines how the plot is mapped onto the screen. If nset = 0 then the default size is used, if nset < 0 then the current viewport is filled. If nset > 0 then cpcnrc does not call SET and assumes the user will.
labs	Determines whether the data high and low points are marked. If labs = 0, then each high is marked with "H" and low with "L". If labs > 0 then each high and low point are marked with their numeric value. If labs < 0 then no high or low points are marked.
dash	Specifies how the contour lines are drawn. If dash = 0, 1, or 1023 then solid lines are used. If dash > 0 (apart from the special values 1 or 1023) then dashed pattern are used. See page 195 of the NCAR Contouring & Mapping Tutorial[1] for an explanation of dashed line patterns. If dash < 0 then dashed patterns are only used for contour lines with negative values.

2.1.3 Area map contours

When contours are set up for an area map (used to define regions for filling contours), then the routine *cprect* is used instead of *cpcnrc*. The parameters for this call are:

call cprect (data, sizex, ncx, ncy, realwork, length_realwork, intwork, length_intwork)

A detailed description of the parameter values to be used when calling *cprect* is contained in Table 2.

Table 2 - CPRECT

Parameter	Description
data	Array containing the data to be contoured
sizex	First dimension of the data array (Not all of the data needs to be contoured, hence ncx can be smaller than sizex)
ncx	Size of dimension of data in the x direction
ncy	Size of dimension of data in the y direction
realwork	Real number workspace array
length_realwork	Size of real number workspace array
intwork	Integer number workspace array
length_intwork	Size of integer number workspace array

The size of the workspace array used in *cprect* has to be estimated according to the amount of data being contoured, and should have a value of at least 10,000, and may be as large as 600,000. Array sizes larger than this approach memory limitations, and the program may not run because it cannot create a large enough memory stack. For most data arrays, a size of 100,000 for the workspace arrays should suffice. The gaseous detonation example, which had a grid of 640 x 640 and initially 18 frames to process, needed workspace arrays of 200,000. If the message:

CPGRWS nnn WORDS REQUESTED, nn WORDS AVAILABLE

is received during execution then the size of the real workspace array needs to be increased. Similarly, if the message:

CPGIWS nnn WORDS REQUESTED, nn WORDS AVAILABLE

is received then the size of the integer workspace array needs to be increased. More information about workspace arrays can be found on page 223 of the NCAR Contouring and Mapping Tutorial [1], (although the numbers they give as estimates are somewhat on the small side).

2.2 Area Filling Routines

One way to improve the effectiveness of the contour plots is to fill the contour areas with identifying colours. The process of defining different areas on the plot is done by storing an area map in memory. The routine *arinam* must be called first to initialise the area map.

After the area map has been initialised and the contouring routines have been initialised (using *cprect*), then the contours can be added to the area map. This is done by calling the routine *cpclam*. To fill the areas with solid colours, the fill style needs to

be set to one, by calling *gsfais(1)*.The routine *arscam* is then called to fill the areas in different colours. The parameters for *arscam* are:

call arscam(map, xwrk, ywrk, nwrk, iarea, igrp, nogrps, fill)

A detailed description of the parameter values to be used when calling *arscam* is contained in Table 3.

Table 3 - ARSCAM

Parameter	Description
map	Area map array
xwrk	X coordinate workspace array
ywrk	Y coordinate workspace array
nwrk	Size of coordinate workspace arrays (same for both x and y)
iarea	Array of area identifiers
igrp	Array of group identifiers
nogrps	Size of iarea and igrp
fill	Area processing and filling subroutine

The coordinate workspace arrays should be the same size as the real and integer number workspace arrays used in *cprect*. The area and group identifier arrays are used when the user wants to mark areas as different types. (This is mainly used when geographic maps are used and the user wants to distinguish between areas over land and areas over water). The nogrps can be set to the minimum, two.

The fill subroutine needs to be defined externally and is called by *arscam* to decide which colour to fill each region. For each group of areas, it determines if the area is defined by three or more points and if so, then each area is filled with the appropriate colour by setting the current colour index to the area identifier and then calling a GKS graphics fill routine. Sometimes the number of the colour may need to be set one or two more than the area identifier number. This is used when background and foreground colours are defined on colour indices 0 and 1, and sometimes 2. For more information on area-fill routines, see pages 48-49 of the NCAR Contouring and Mapping Tutorial [1].

The following code outlines the area map filling calls used in the bullet impact model.

```
C       Initialize Areas
              call arinam(map, lmap)
C       Turn off automatic contouring and set number of contour levels
              call cpseti('CLS - Contour Level Selection Flag',0)
              call cpseti('NCL - Number of Contour Levels', numcont)

C       Set the contour level values and set contour colour to 1 - black
        do 300, i=1, numcont
              call cpseti('PAI - Parameter Array Index', i)
              call cpsetr('CLV - Contour Level Values', contlevel(i))
              call cpseti('CLC - Contour Line Colour Index',1)
 300    Continue

C       Initialise contour map
              call cprect(prs, ncx,ncx,ncy, rwrk, lwork, iwork, liwork)
C       Add contours to area map
              call cpclam(prs, rwrk, iwork, map)
C       Set fill style to solid, and fill contours
              call gsfais(1)
              call arscam(map, xwrk, ywrk, nwrk, iarea, igrp, nogrps, fill)
C       Draw Perimeter
              call gridal(64,0,64,0,0,0,5)
```

The size of the area map used in *arscam* has to be estimated from previous experience. In the gaseous detonation example, an area map array of 600,000 elements was needed. Generally area maps should range upwards from 100,000, and the size of the area map should be larger than the workspace arrays. If the message

ERROR 5 IN AREDAM - AREA-MAP ARRAY OVERFLOW

is received during execution then the size of the area map array needs to be increased. A discussion of area map arrays is found on page 27 of the NCAR Contouring and Mapping Tutorial [1].

2.3 Cell Arrays

Another method of producing colour contour plots is to use cell arrays. This method involves setting up a two dimensional array (the cell array), which stores the colour representing the value of each cell in the finite difference grid. On a typical 640 x 640 finite difference grid, for example, the cell array would also normally be dimensioned to 640 x 640, even though, for viewing purposes, the resolution on the PC screen may only be 640 x 480.

When data is being read into memory, the user can compare the data values to user-defined thresholds. If the data is above a certain level, then a particular colour is stored

in the cell array. The cell array has advantages over the contouring method in that it is more accurate than the contouring method, and usually takes less CPU time to generate. The screen is filled with colours from a cell array by calling *gca*.

The code below shows an example of data (in the array *prs*) being compared to user-defined thresholds (in the array *range*) and the appropriate colour index being stored in the cell array.

```
290             if (col.lt.colnum) then
                        col = col + 1
                        if(prs(i,j).lt.range(col)) then
                                colia(i,j) = col
                        else
                                goto 290
                        end if
                end if

C       Display the cell array
                call gca(0.0, 0.0, 1.0, 1.0, ncx,ncy,1,1,ncx,ncy,colia)
```

The general parameters for a call to *gca* are:

```
call gca(px, py, qx, qy, dimx, dimy, isc, isr, dx, dy, colia)
```

A detailed description of the parameter values to be used when calling *gca* is contained in Table 4.

Table 4 - GCA

Parameter	Description
px	World coordinate giving x value for first corner point of rectangle
py	World coordinate giving y value for first corner point of rectangle
qx	World coordinate giving x value for second corner point of rectangle
qy	World coordinate giving y value for second corner point of rectangle
dimx	First dimension of colour cell array index
dimy	Second dimension of colour cell array index
isc	Index of starting column to be used in colour cell array index
isr	Index of starting row to be used in colour cell array index
dx	Number of columns to use from colour cell array index
dy	Number of rows to use from colour cell array index
colia	Array of colour indices dimensioned dimx by dimy.

For more information about cell arrays see pages 388-389 of the Contouring and Mapping Tutorial [1], and pages 161-163 of the NCAR Fundamentals Manual [2].

2.4 Colour Maps

NCAR uses a red/green/blue (RGB) colour scheme. This is an additive colour scheme, that is, colours are defined as containing amounts (a real value ranging between 0.0 and 1.0) of each of the primary colours. For example, white is represented by the values (1.0, 1.0, 1.0), yellow by (1.0, 1.0, 0.0) and light blue by (0.5, 0.8, 1.0).

When NCAR performs any operation that involves writing to the screen, it displays the object in a colour represented by a colour index for that type of item (eg. there is a colour index for contour lines, for screen backgrounds, for area fills, etc.) These colour index numbers need to be defined prior to their use. Each number in the index is given an appropriate set of RGB values. The simplest way to implement this is to have a separate colour subroutine that is called early in the program.

The choice of a suitable colour scheme is an important consideration. The scheme has to give an indication of the gradual rise in the variable being contoured, but the colours have to be different enough to allow the viewer to easily distinguish between adjacent colours, and hence highlight the areas of interest. The number of colours to be used is also a consideration. The software connecting the PC to the workstation in this work could only display approximately 200 colours at one time, while the viewing software on the workstation is restricted to approximately 180 colours.

For the basic area filling contour plots 16 colours were used, enough to allow clearly distinguishable colours. White or black was set as the colour representing the lowest contour level, depending on the background colour, and then light blue, dark blue, green, yellow, orange and red represented successively higher contour levels. This colour scheme is illustrated in Figure 1a.

Another way of displaying the features of the flowfield is to use the cell array method, which was described in the previous section. In this method considerably more colours are used, and to avoid having to define each colour index and set of RGB values manually it is a simple matter to set up a loop that starts off with a colour, such as white (1.0, 1.0, 1.0), and moves in fixed increments towards a different colour, such as red (1.0, 0.0, 0.0). This is achieved by decrementing the current values of the blue and green index during each cycle of the loop, and creating a colour which is closer to red. We used a total of 161 different colours for the cell array method, and these are illustrated in Figure 1b.

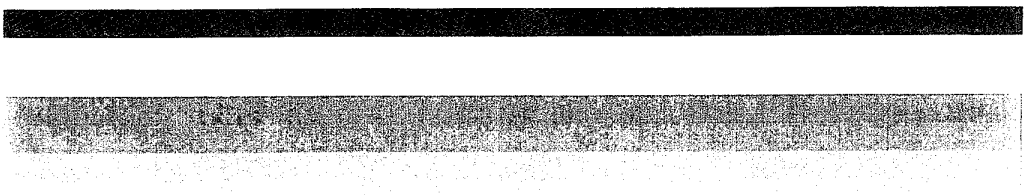


Figure 1(a): Colour scheme used for basic area filling contour plots.

Figure 1(b): Colour scheme used for plots produced using the cell array method.

When colour plots are produced using either the colour contour method or the cell array method, some care is needed to ensure that the full range of colours appears in the plot. For example, in the gaseous detonation example described in section 3, if the full range of numerical values is partitioned by a fixed interval, then several colours may not be present in the plot because they correspond to numerical values which are within the range, but which are not present in the data. To avoid this problem, it is necessary to distort the range of the colours, so that for example five colours are used to display values between 100 and 200 (values that don't occur frequently) and 35 colours are used to display numbers between 0 and 99. This has the effect of providing enough contrast in important regions of the flow field. An example of the coding for this type of approach is shown in the plotting routine listed in Appendix 9.3.

2.5 Label Bars

There are a number of NCAR routines that can be used to add extra information to the plots. Label bars are a colour key for the plot, that is, they show what numerical value each colour represents. Label bars are implemented by using an array of colours and a matching array of characters. The numerical values associated with each colour need to be written into the array of characters and then they will be displayed on the screen.

Label bars are produced by calling the routine, *lblbar*.

```
call lblbar(ihov, xmin, ymin ,xmax, ymax, nbox, wsfb, hsfb, colors, iftp, lbls,
numlbls, lbab)
```

Titles, background grids, and other labels can easily be added to the plot. The label is positioned on the screen, and then the appropriate routine is called. The program listings in the Appendix provide several examples. For more information about label bars see pages 189-194 in the NCAR Fundamentals Manual [2].

A detailed description of the parameter values to be used when calling *lblbar* is contained in Table 5.

2.6 Aspect Ratio

NCAR automatically resizes the plot to a square that will take up the maximum amount of room on the screen. This however can distort the picture, and for some plots it is important to maintain a correct aspect ratio. There are two different routines which can be called to do this.

The *set* routine sets the position and size of the plot on the screen. For examples showing the use of *set* refer to the code listings in the Appendix, and more information about *set* can be found in the Contouring & Mapping Tutorial [1], pages 216-217, and the Fundamentals manual [2], Chapter 8.

Set is called in the following manner:

call set(a1,a2,b1,b2,1.0,rx,1.0,ry,1)

A detailed description of the parameter values to be used when calling *set* is contained in Table 6. The 1.0 indicates the values of the left and bottom of the viewport, and the 1 indicates a linear - linear plot.

Note that in some cases, some of the contouring routines call their own version of *set*, which will override user-defined settings. To avoid this, the parameter 'SET' should be given the value 0, using *cpseti*.

call cpseti('SET', 0)

The other call to directly set the aspect ratio is to change the parameter VPS using *cpsetr*.

call cpsetr('VPS',-(a2-a1)/(b2-b1))

where a1,a2,b1,b2 are calculated the same way as for SET.

Table 5 - LBLBAR

Parameter	Description
ihov	Flag indicating whether the label bar should be printed horizontally or vertically. 0 is used for horizontal & 1 for vertical. (Note, the manual incorrectly lists 0 as representing both horizontal and vertical).
xmin	Real number between 0.0 and 1.0 representing left coordinate position of label bar.
xmax	Real number between 0.0 and 1.0 representing right coordinate position of label bar.
ymin	Real number between 0.0 and 1.0 representing bottom coordinate position of label bar.
ymax	Real number between 0.0 and 1.0 representing top coordinate position of label bar.
nbox	Number of boxes into which the label bar should be divided.
wsfb	Real number between 0.0 and 1.0 representing fraction of the width of each box to be filled.
hsfb	Real number between 0.0 and 1.0 representing fraction of the height of each box to be filled. (Note, if the box is completely filled, the numeric value of the box will be written over and disappear from view).
colors	Array of colour indices for the box.
iftp	Specifies how the box should be filled. Set to 1 to indicate default filling pattern.
lbls	Character array of labels for each box. The numeric values of the contour levels are read into this earlier in the contouring program.
numlbls	Number of labels to be used.
lbab	Specifies position of labels. Set to 1 to get labels underneath boxes.

Table 6 - SET

Parameter	Description
a1	a1 is the left edge x coordinate position of the contour grid. It should usually be set to 0.05
a2	a2 is the right edge x coordinate position of the contour grid. It should usually be set to 0.95
b1	b1 is the bottom edge y coordinate position of the contour grid. It should usually be set to 0.05
b2	b2 is the top edge y coordinate position of the contour grid. It is calculated by $b2 = b1 + ((a2 - a1) * (ry \div rx))$
rx	The dimension of the grid in the x direction
ry	The dimension of the grid in the y direction

2.7 Contouring Selected Regions

It is possible to contour selected parts of the grid of data only. This is useful when a certain part of the grid needs to be examined in more detail. If the first portion of the array is all that is needed to be viewed then the call to cprect can specify the amount that is needed using ncx and ncy. However if a segment in the middle region of the array is wanted, then this segment needs to be copied into a new array. This new array is then contoured using either the area filling method or the cell array method.

2.8 Printing

The gmeta files are formatted in a device independent standard graphics format which, if they are to be printed, need to be converted into a format recognisable by a printer. The files can be converted into a range of printer languages, with postscript being the preference within WSD-M. This is done with the NCAR command *ctrans*, issued at the UNIX prompt.

ctrans -d ps.color gmeta > filename.pscolor

To print to the WSD-M Phaser colour printer, the postscript file has to be transferred across to the network disk. Then, using the computer in laboratory 6, to which the colour printer is attached, the file can be printed with the following DOS command:

copy filename lpt1:

This transfers the file through the computer's parallel port directly to the printer.

To print to the colour HP DeskJet 550c, the postscript file needs to be converted into a format that the DeskJet can read. This is done by running the *GhostScript* program which is available on the HP's. An example of the UNIX commands follows:

gs -sOutputFile=filename.dj550c -sDEVICE=cdj550 filename.pscolor
lp -d dj550c -or filename.dj550c

Note that the spacing and capitalisation should be exactly as shown above.

3. Examples

Using the NCAR routines described in the previous section, a standard framework has been developed to produce colour contour plots of the data generated by the AMRL hydrocodes. By changing appropriate parameters, and retailoring the colour scheme, this can be adapted for use with different data sets. This section illustrates the application of these routines to selected areas of work currently in progress in WSD - M. These include recent numerical simulations of projectile impact on the explosive Composition B, using the 2D Eulerian multimaterial FCT (Flux Corrected Transport) hydrocode currently under development in WSD, and highly resolved 2D simulations of the behaviour of gaseous detonation fronts in a mixture of $H_2/O_2/Ar$. Individual colour plots can be viewed on either the HP Workstation or on the PC (using *X Vision*). These can be printed in colour using the division's colour printer (Tek Phaser II SDX), or the HP colour DeskJet (DeskJet 550C).

3.1 Bullet Impact

WSD is currently studying the effect of projectile impact on the explosive Composition B using flat nosed cylindrical copper projectiles [3]. For high projectile velocities the explosive detonates and releases energy at an extremely fast rate. At lower velocities the response is less violent, energy is released at a much slower rate, and a deflagration, or very rapid burn, occurs. The experiments currently being conducted in WSD are designed to determine the threshold velocity for detonation, and then to examine the effect of the physical composition of the explosive on the threshold velocity.

These experiments are currently being modelled numerically using a multimaterial 2D Eulerian hydrocode under development in WSD [4,5]. The multimaterial nature of the code means that in addition to plotting the variable contours for each material in the problem, additional calls must be made to outline the boundaries of each material. For example, in the bullet impact problem there are three different materials in the grid;

the bullet, the explosive, and the surrounding air. The ability to outline the boundary of each material on the contour map is quite important. Showing the outline of each material allows the viewer to see how the force of the explosion has deformed the confining material, and also shows the effect which the confinement has upon the detonation.

The method used to overlay the interface between the different materials in the grid was to contour the data files containing the volume fractions of each material at each grid location. The volume fractions range between 0.0 and 1.0, and so by selecting a single contour level, say 0.5, an outline of the selected material can be produced. This is then repeated for each material in the grid. This procedure has to be followed after the area filling routines have been called, otherwise the contour lines will be overdrawn. This procedure is illustrated in the program listing contained in Appendix 9.1, which produced the colour plots shown in Figure 2.

Figures 2a and 2b show the pressure contours within the explosive (Composition B), a steel bullet, and the surrounding air after 1200 cycles, or approximately 4.0 μs. In Figure 2a the bullet was travelling with a velocity of 1100 m/s and the impact has resulted in a detonation within the explosive. In Figure 2b the bullet was travelling with a velocity of 900 m/s, and this has resulted in a deflagration.

The simulation was conducted in cylindrical geometry and the grid contained 130 cells along the axial direction, 45 cells along the radial direction, with $\Delta x = \Delta y = 0.033$ cm. The explosive initially occupied the region defined by $x = 41$ to $x = 130$, $y = 1$ to $y = 45$. The bullet initially occupied the region defined by $x = 11$ to $x = 40$, $y = 1$ to $y = 15$, and the rest of the grid was filled with air. The single black contour in Figures 2a and 2b clearly shows the outline of the bullet, and the explosive/air interface. The plots were produced using the colour contour method.

3.2 Gaseous Detonation

For the past few years, WSD has been using numerical simulation techniques to study the transfer of detonation between explosive layers [6-8], and the failure and reignition of detonation behind reflected shocks [9-11]. Recently, these studies have been extended to consider the more general problem of the reignition of detonation at an abrupt area change [12].

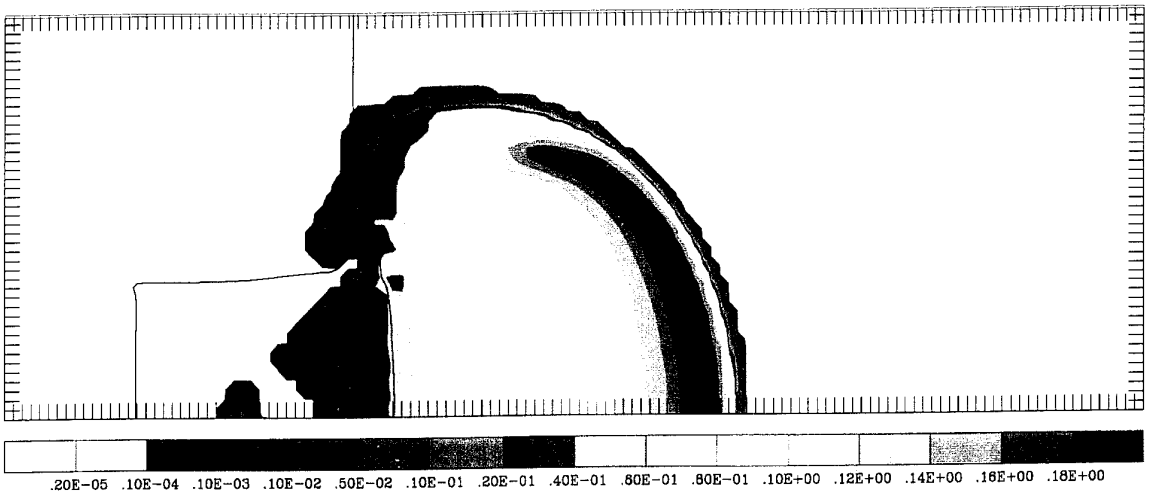


Figure 2(a): Contour plot for the bullet impact simulation showing pressure levels within the Composition B explosive, steel bullet, and surrounding air. The bullet was travelling with a velocity of 1100 m/s and the colour plot shows that the impact has resulted in a detonation.

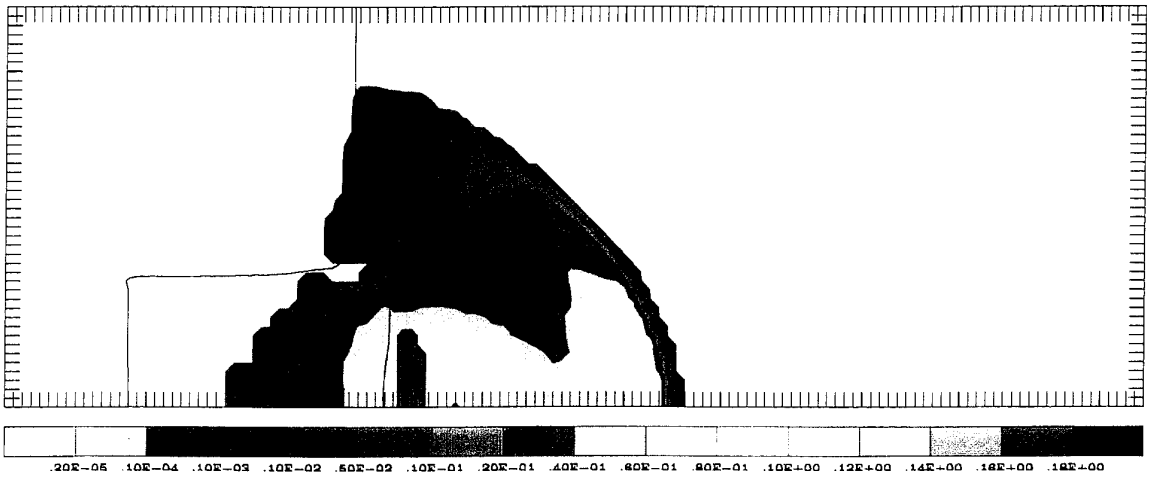


Figure 2(b): Contour plot for the bullet impact simulation showing pressure levels within the Composition B explosive, steel bullet, and surrounding air. The bullet was travelling with a velocity of 900 m/s and the colour plot shows that the impact has resulted in a deflagration.

A detonation wave is a supersonic shock wave travelling through a reactive medium. The high temperature generated by the passage of the shock through the material initiates chemical reaction and energy release. A stable detonation with a unique detonation velocity will develop if the chemical reaction zone can remain coupled to the wave and continuously feed energy to the shock front. The mechanism by which the released energy is fed to the shock front, and the nature of the shock front itself, is quite complex. On a microscopic scale the detonation front has a highly three-dimensional, nonsteady, cyclic behaviour characterised by the presence of transverse waves propagating perpendicular to the direction of the detonation front, and forming a characteristic "cellular structure".

Figure 3 shows a schematic of a two-dimensional detonation front at selected times, showing the cyclic nature of the detonation front and the presence of the transverse waves. The interaction of the transverse waves with the detonation front results in a complicated shock interaction and the formation of a third shock known as a "Mach stem". The pressure and temperature behind a Mach stem are very high, so chemical reaction takes place rapidly and the rate of heat release is high. This "mini explosion" results in the generation of new transverse waves, which then move out along the detonation front and collide with neighbouring transverse waves, creating new Mach stems and mini explosions as the old ones die out. This quite complicated process is the mechanism by which the detonation front is sustained, and a detailed model of the interacting shocks is required to fully understand the spontaneous reignition of detonation which can occur at an abrupt area change.

Figure 4 shows a schematic of a detonation front expanding into a larger diameter tube. In the upper part of Figure 4a the cellular structure of the detonation front is clearly discernible, while in the lower part of the diagram the cooling introduced by the expansion has caused the shock front and the reaction front to become separated, and the detonation has begun to die. In Figure 4b however a transverse reactive shock has formed, and this will eventually lead to the reignition of the detonation.

Recent work in WSD has attempted to understand this "spontaneous reignition" process in more detail by simulating the expansion of a detonation wave travelling in a Hydrogen(H_2)/Oxygen(O_2)/Argon(Ar) (2:1:7) mixture into a larger area [12]. The calculations are performed using a 2D cylindrical finite difference hydrocode which simulates the propagation of a detonation travelling in the upper part of a double barrelled detonation tube, and then the expansion of the detonation as it reaches the end of the dividing splitter plate. Figure 5a shows the pressure contours from one of these calculations after 3600 time steps, or a time of 72 μs. The calculation uses a 640 x 640 grid with $\Delta x = \Delta y = 0.04$ cm and $\Delta t = 0.02$ μs. The detonation was initially confined to the upper left of the grid by a steel splitter plate (not shown), and has subsequently diffracted around the edge of the plate. The detonation cells on the leading front of the detonation are clearly visible (upper right), and the formation of new cells on the curved lower part of the front (lower right) show that the detonation has reignited, and will subsequently generate a new front spanning the enlarged diameter of the tube.

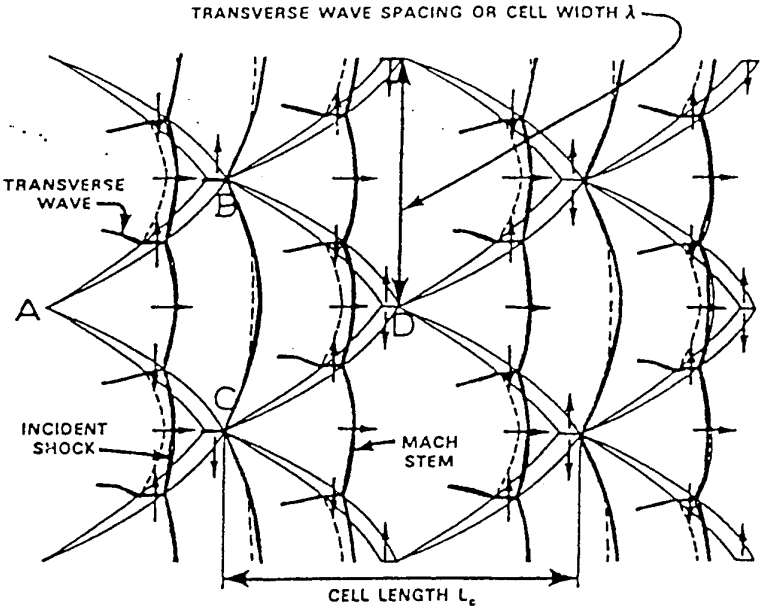


Figure 3: Schematic of detonation front showing transverse wave structure.

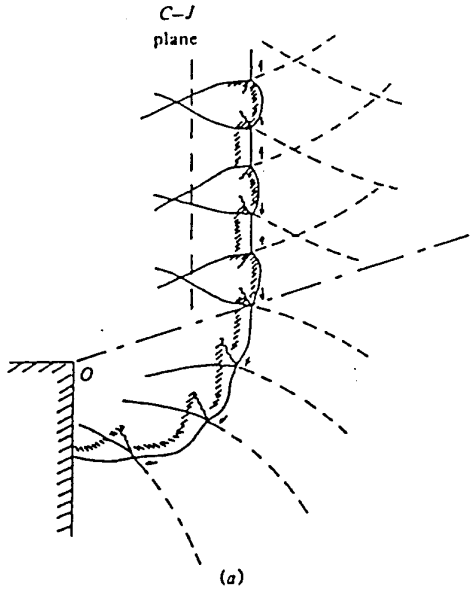


Figure 4(a): Schematic of detonation front expanding into a larger diameter tube. Separation of the shock front from the reaction front in the lower part of the diagram indicates that the detonation is failing.

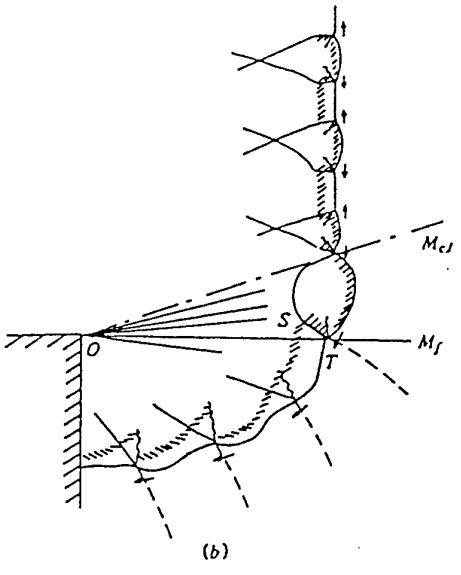


Figure 4(b): Schematic of detonation front expanding into a larger diameter tube. The strong detonation cell in the centre of the picture formed by the transverse wave structure indicates that the detonation will eventually reignite.

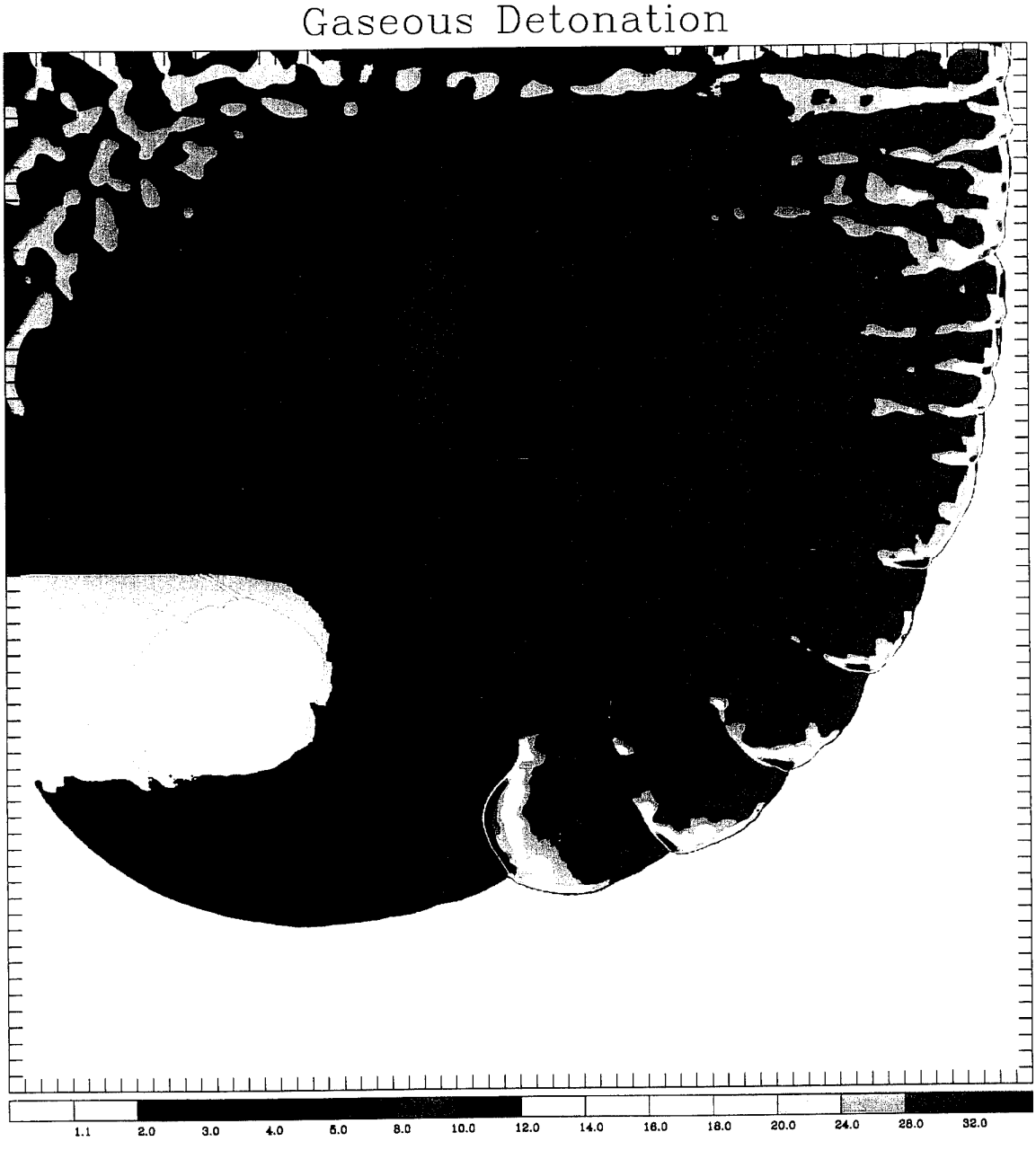


Figure 5(a): Simulated pressure contours showing reignition of detonation at an area expansion. The area map filling and contouring scheme was used to indicate the various pressure levels.

Figure 5(b): Simulated pressure contours showing reignition of detonation at an area expansion. The cell array method was used to indicate the various pressure levels.

Figure 5a was generated using the area map filling and contouring scheme discussed in section 2.2, and the FORTRAN program which produced the plot is listed in Appendix 9.2. A different way of plotting the data however is to use the cell array method discussed in section 2.3. This results in the plot shown in Figure 5b. This contains more detail, but the added detail results in a larger file size (33 megabytes, as opposed to 2 mbytes for Fig 5a), and the plot takes longer to produce on the colour printer (one and a half hours as opposed to 5 minutes). The program which produced Fig 5b is listed in Appendix 9.3.

3.3 Triple Point Trajectories

In a detonation front, the point at which the lead shock, the transverse wave, and the Mach stem collide is known as the triple point. For a regular detonation, the paths of the triple points trace out a diamond like structure, which can be seen experimentally if a thin layer of soot is applied to the inside of a detonation tube. Figure 6 shows a schematic of the diamond like structure produced on the walls of a detonation tube by a regular detonation. The ability to trace these movements is desirable because they give an indication of whether a detonation is regular or irregular. It is difficult to ascertain this otherwise, unless a series of plots depicting different times are examined.

To produce a plot of the triple points it was necessary to examine the selected variable at each grid location and at each time step, and if the value was above a set threshold value, to mark that location using a colour array. The colour array was initialised so that each location was set to the background colour (usually white), and if a location was marked then the corresponding element in the array was given a colour index (usually black) to provide contrast with the background.

This process continues for all of the time steps. However the routine *frame*, which is usually called after each time step to produce a plot, is not called until the calculation has finished. This ensures that there is only one plot produced, and that it has the information from all of the time steps - effectively tracing the path of the highest value regions. Care must also be taken not to reset the colour array at each time step, it should be initialised outside the main loop.

Some experimentation is involved in setting the threshold value which determines which regions are marked as triple points. Several values were trialed to see which produced the best results. As the threshold is made higher, then less points are marked and the pattern becomes less detailed. If the threshold is lowered, then more detail is added until the pattern becomes blurred by overwriting.

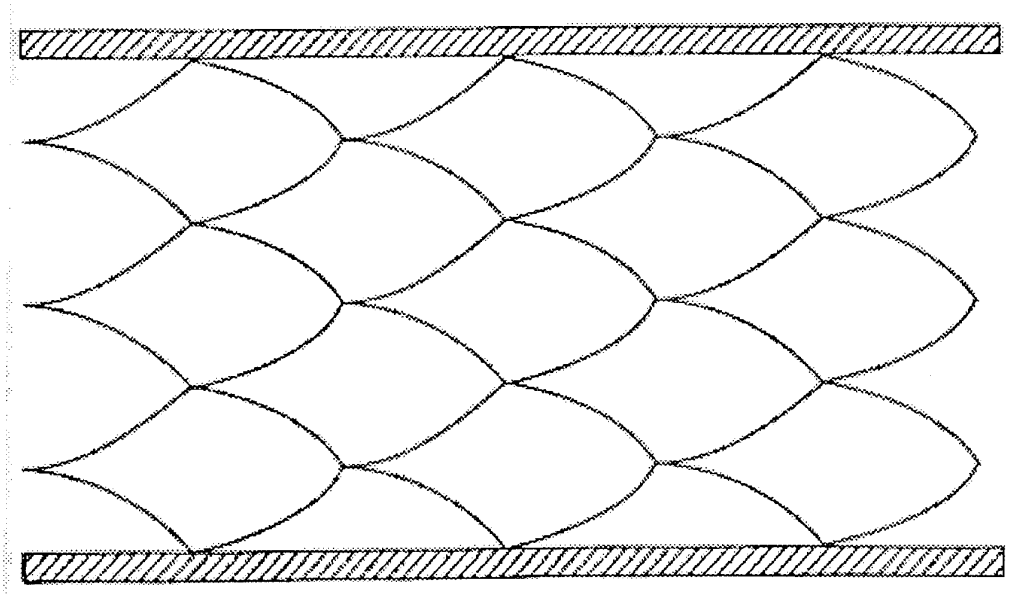


Figure 6 - Schematic diagram of the triple point trajectories produced by the cellular structure in a regular detonation front.

The routines to mark the path of the triple points were incorporated into the hydrocode model of a gaseous detonation, and the code was run on the relatively simpler problem of an initially planar detonation propagating along a constant diameter channel. The simulation was run in planar geometry with 960 cells in the direction of propagation of the detonation, and 80 cells in the transverse direction. Initially a planar detonation was placed on the grid, and then this was perturbed by placing a high pressure region in front of the detonation. This resulted in a complicated sequence of shock interactions, but after approximately 96 μs (4800 time steps) a steady pattern developed. Figure 7 shows the resulting triple point motion in the section of the grid between x = 480, x = 800, and y = 1, y = 80. Figure 7a shows the pattern produced by the temperature if the threshold value is set to 3000K, while Figures 7b and 7c show the patterns produced by the density and pressure, with threshold values set at 3.3 kg/m^3 and 25.0 x 10^5 Pa, respectively. The pattern shows a highly regular behaviour, with four detonation cells spanning the width of the tube. The section of FORTRAN coding within the main detonation program which produced this plot is listed in Appendix 9.4.

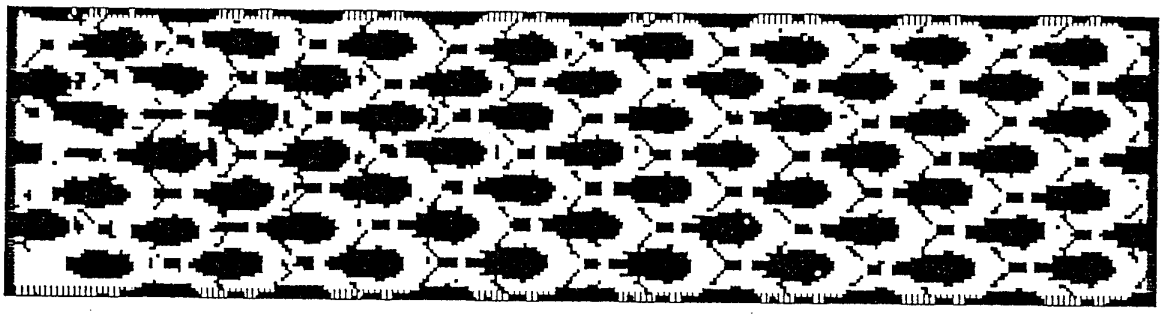


Figure 7(a): Triple point trajectories produced from temperature plots.

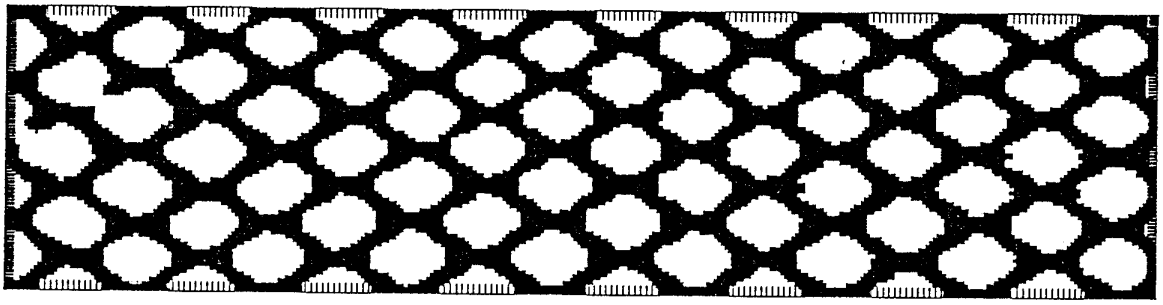


Figure 7(b): Triple point trajectories produced from density plots.

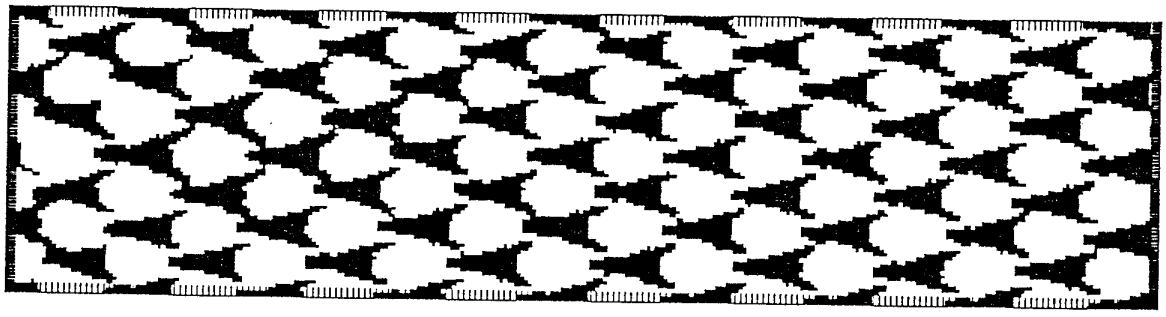


Figure 7(c): Triple point trajectories produced from pressure plots.

4. Video

This section looks at several different methods for taking a series of plots and transferring them to video tape. The methods described here were determined by the hardware and software currently available within AMRL. The following section will examine improvements which could be made if other facilities were available.

4.1 Equipment - Scan Converter/Video Card

An item of equipment is required which is able to transfer the contents of a computer screen to video. This can be either an external scan converter which is connected to both the computer and video recorder, or an internal video output card, which is slotted into the computer and is able to output the computer's display in a format that can be recorded by the video. To produce the video sequences described later in this report an external scan converter was borrowed from the Photography & Applied Imaging Section of AMRL. The scan converter (Yamashita Engineering Manufacture Inc., model CVS-910) converts VGA signals from a PC into composite video format. The scan converter was connected to a PC on the divisional LAN network, and the output from the scan converter was then connected to a standard VHS video recorder. Subsequently a Super VHS video recorder was used to improve the quality of the recording.

The scan converter has the ability to focus on particular areas of the screen, being able to zoom up to 4 times the size. This means that it is possible to record only part of the screen, allowing the user to have other software running (such as an animator or plot viewer) in other areas of the screen without them appearing on the video.

As all the calculations and plotting were performed on the HP workstations, the plots had to be transferred to the PC. This was done using the X-terminal emulator *Xvision*.

4.2 X Vision

Using the software package *XVision*, it is possible to view plots from the workstation on a PC running *Windows*. It does this by emulating XWindows on the screen and establishing a connection to the Workstation. Displaying the plots on the PC screen allows the scan converter to write them to video, and also allows the use of a screen capture program to save the plots in a file format recognised by the PC (for example bitmaps, .bmp files).

An important point to note when using *XVision* is that it fails to refresh the screen if another window temporarily covers the plot window. The plot underneath will have been erased. It is possible to redisplay the plot, but if the plot was lost when in the

middle of capturing screens, and there are 720 frames, then a lot of work is needed to go back and capture that particular frame.

4.3 Writing Frame by Frame to Video

The simplest practical way to transfer a sequence of plots from the PC screen to videotape is to display them in sequence on the screen, and then to press pause on the video between each plot. This gives the PC time to refresh the screen. Either the *ctrans* or idt facilities (part of the NCAR software) can be used to cycle through the frames. *Ctrans* is invoked by entering *ctrans -d X11 gmeta* at the UNIX prompt. This allows the user to move from frame to frame, but only in the forward direction. *Idt* is the preferred option because it allows the user to move in both directions through the sequence, which is a useful feature to have in case a frame is missed. It is invoked by entering *idt*

The disadvantage of this method is that it takes some time for the video heads to adjust between pausing. The VCR used was unable to advance a frame at a time, which meant that a small amount of time (approximately one second) had to elapse to ensure that the VCR did not overwrite the tape. This means that there will be one frame shown per second, which is generally too slow for an animation (normal animation standards are between 25 and 30 frames per second). Repeatedly pressing pause can also cause the image to flicker.

4.4 Video for Windows

The preferred method used was to capture each plot using a screen capture program and to use *Video for Windows* to create an animation on the PC, which was then recorded onto the video.

Two different screen capture programs, *Grabit* and *WinCapture*, were used. Both programs work in the Windows environment. To use *Grabit*, the *Grabit* window is minimized and then the desired plot is shown on the screen. The *PrintScreen* key is then pressed, storing an image of the screen in the Windows Clipboard, and then *Grabit* is reopened. Using *Shift-P* the screen in the Clipboard is then pasted into *Grabit*, and then the screen can be saved as a bitmap. The next plot is then displayed on the screen, and the process is continued. This process captures the whole screen, so when the animation is being run the window bars and NCAR graphics sign will appear. However, by adjusting the area onto which the scan converter is focussed, these can be ignored.

In the capture setup menu of *WinCap* the user specifies the hot key that activates the process of grabbing a screen and where the files are to be stored. The user should specify that they want to grab the window's client area; this is the part of the NCAR window which contains the plot. *WinCap* also allows the user to manually define the

area to grab each time. This means that a desired portion of the NCAR window can be grabbed. However it is difficult to get precisely the exact same size every time, meaning that when the sequence is animated, it tends to flicker around the edges.

The plots are displayed using either *ctrans* or *idt* and for each frame, the hot key is pressed. A copy of the NCAR window is then saved with the filename cap-nnn.bmp, where nnn is one more than the number of the last bitmap saved in the chosen directory.

WinCap also has the property that it may turn areas of white background in the NCAR plots to black. Depending on the type of plot, this may or may not be a problem. For the bullet impact models this was not acceptable, as it hid the position of the bullet (marked in black lines) from view. Hence *Grabit* was used to capture the bullet impact frames.

The disadvantage with this method is the large amount of time involved in capturing the screens one by one, particularly when the number of frames is as large as 720. There is also the possibility of missing one or more of the frames in such a long sequence, and if *ctrans* is being used to display each plot, then a lot of effort will be required to capture the frames which have been lost.

Once all of the frames have been converted into bitmaps they can then be inserted into a video sequence, or AVI (Audio Video Interleaved) file. This is done using *VidEdit* (part of the *Video for Windows* package). (*VidEdit* had to be installed in a local *Windows* on the PCs hard drive because it tries to make changes to the system files which the system *Windows* won't allow.) Each bitmap is inserted in order. The number of bitmaps that can be fitted into a sequence in the editor depends on the memory of the computer. Using a PC with 20 megabytes of RAM, the maximum was slightly more than 150 in one file. The machine tended to slow down if anything like this number was used however, and so editing was done in blocks of 50. At a later stage of the work a PC with 32 megabytes of RAM was used, and then 300 frames could be loaded into one sequence without significant loss in speed. This limit applies only when editing the video sequences. A video sequence that has been saved with 300 frames can be played back on a machine with 20 megabytes of RAM.

Once this capability was determined manually, an automated Windows Macro was developed. In this procedure *ctrans* was used to display the plots frame by frame on the PC screen through Xvision. (*idt* was unable to be used as it overwrote the plot window, making capturing impossible). The *WinCap* program was set to capture the plot window to the clipboard, and then this information (frame) was pasted directly into *VidEdit*. This means that there is no need to save each frame as a bitmap, but care should be taken in the fact that only approximately 80 bitmaps can be pasted into *VidEdit* before memory problems occur. Hence it is necessary to break the sequences into 80 frame lots. Once converted into AVI files they can then be combined into one long sequence using *Video for Windows*, subject to memory limitations.

Once the files have been saved as AVI files, they can be loaded into *Media Player* (part of the standard *Windows* accessories). To help the animation run smoothly, one of the options that must be set on *Media Player* is the command that stops it from skipping frames. In the *Device* menu (which only becomes active when a file is loaded), the option *"Configure"* should be chosen. This displays the default settings for *Media Player*. The option *"Skip video frames if behind"* should not be marked. Once this option has been turned off, then these settings should be saved as the defaults. *Media Player* will then keep these defaults and will not skip frames.

If the number of frames in the sequence is relatively small, then each AVI sequence can be combined into one long AVI file by using VidEdit. This can then be recorded onto the VCR at once. This was able to be done for a sequence containing 720 frames using a 486 DX-66 with 32 Mbytes of RAM. Alternatively, if the number of frames is unable to be stored in one AVI sequence, then as the end of each file in the AVI sequence is reached the video is paused, the next file in the sequence is loaded, the video is unpaused, and the new file is started. This procedure may result in a slight jump in the animation at this point, unless some care is taken to ensure the correct timing of the pause button.

The files were loaded, played and recorded onto video. At this stage however the PC was struggling to refresh the screen quickly enough, and the total sequence of 720 frames took approximately seven minutes to run. This was too slow to be of use at conference presentations, or to be helpful in watching the evolution of transient events in the flow field.

To increase the speed of the animation, the sequence was run on a faster PC. The original PC used was a 486-33 Mhz with a standard SVGA card. The faster PC was a 486-66 Mhz with a local bus architecture and a *Windows* accelerator card. The video sequence took slightly under two minutes to run on this machine. This was close to the desired speed, and so the final copy of the tape was made using the faster PC.

Note that the PC speed is only of secondary importance in transferring the animation sequence to video. Of critical importance is the type of video card and bus architecture. An ISA bus (used in the 486-33 machine) has a transfer speed of approximately 8 MB/s, whereas a VESA local bus (used in the 486-66 machine) can transfer up to 32 MB/s. In this sequence we are transferring 720 frames, which at 25 frames per second should take approximately 28 seconds. Using the ISA bus the elapsed time was approximately 420 seconds, and with the VESA local bus it was approximately 120 seconds. Consequently, if we want the video sequence to play at the correct speed we would require either a much faster transfer rate, or a video recorder which can record frame by frame and which can ideally be controlled by the PC doing the transfer.

An alternative way to increase the speed of the video is to reduce the number of frames, but this is not an acceptable option as it will tend to make the animation more jerky, as the front will be moving greater distances between frames.

If the PC used to play back the video sequence has a different resolution than the PC on which the plots were recorded as bitmaps (eg. the original machine was set up with a 640 x 480 resolution and the play back machine has a 800 x 600 resolution) then the AVI frame will not take up the whole screen. Care must be taken not to move the position of the *Media Player* between the different files in the sequence being played, otherwise the position that the AVI file appears in will change, and the next part of the sequence will be in a slightly different location.

4.5 Video Output

A five minute video sequence was made to illustrate the techniques just described. An opening sequence, including titles, was prepared using the *Macromedia Action 2.0* presentation software .

The first part of the video shows output from two bullet impact simulations which were run on the Workstation, one resulting in a detonation, the other in a non-detonation. Both calculations produced approximately 40 time frames and were converted into colour plots using the colour contouring method. They were then edited into two animation sequences using *Video for Windows* and written onto video. Both displays show a 600 x 100 grid of data and each video takes slightly less than one minute to run.

The second part of the video shows the output from a simulation of the expansion of a gaseous detonation as it reaches the end of a splitter plate in a detonation tube. The calculation was run on the WSD Workstation and produced 720 frames of data. These were converted into colour plots using the cell array method, and were then edited into an animation sequence using *Video for Windows*. The display shows a 640 x 640 grid of data and the video takes approximately two minutes to show the detonation as it moves from an initially confined state, to the final expanded and reignited state.

The bullet impact model was recorded on VHS, while the master copy of the gaseous detonation animation was made on a Super VHS tape to maximise the quality of the recording. The video editing facilities in the Photography & Applied Imaging Section were then used to copy the gaseous detonation animation onto the VHS tape containing the bullet impact animations. These tapes were recorded in PAL format, but they can subsequently be converted to NTSC format if needed. A copy of the video tape is available from the authors.

5. Limitations and Equipment Requirements

5.1 Memory

The NCAR program which uses the colour contour method to produce a colour plot of the gaseous detonation requires a large amount of memory to run. For example a Hewlett Packard workstation (EOD-HP1) with 64 megabytes of RAM had insufficient memory to store 18 frames of a 640 x 640 grid in one array, and so the program had to be run on a HP2, which has 96 megabytes of RAM. A way around this problem is to partition the dump file into several smaller files and then to plot each file separately. In this way the size of each gmeta file is reduced, and less data has to be held in memory at any one time.

When editing the video sequence it is desirable to minimise the number of AVI files needed, as the video will need to be paused between loading the next AVI file in the sequence. The size of the AVI files is limited by the amount of frames *VidEdit* can hold in memory at one time. Using the 486DX-33 with 20 megabytes of RAM, approximately 150 frames could be held in memory and saved as an AVI file. With the 486DX-66 with 32 megabytes of RAM, at least 300 frames could be held in memory. An alternative method is to combine the AVI files into one using *VidEdit*. Using this method a sequence of 720 frames of a 640 x 640 grid could be held in memory.

5.2 Disk Space

Large amounts of disk space are needed to store the data files, the gmeta files, and the bitmaps of each frame. The size of the data files that contain the numeric values of a chosen variable depend on the size of the grid, and the number of frames, or time steps involved. Files can vary in size from 1 or 2 megabytes up to 180 megabytes for a file containing 18 time steps on a 640 by 640 grid.

The gmeta files produced store the plots for each frame. Using the contouring method, file sizes vary between half a megabyte, up to 5 megabytes for 18 frames of a 640 by 640 grid. Using the cell array method means that the gmeta files are considerably larger. The final gmeta file that contained 720 frames of a 640 by 640 grid was 303 megabytes in size.

Capturing each frame using a screensaver means that each frame is saved as a bitmap. The 720 bitmaps took up approximately 212 megabytes of disk space. The video sequences of frames are stored as AVI files. To hold the whole sequence of 720 frames takes approximately 11 megabytes. If the automated procedure is used then the bitmaps do not need to be saved and the only storage requirements are for the AVI files.

5.3 PC Resolution and XVision

The 486 PCs have lower resolution than the HP workstations. The workstations have a resolution of 1280 x 1024, while the PC resolution depends on the VGA card installed. SVGA can go up to 1280 x 1024, but in most cases they will then only display 16 colours. The 486DX-33 MHz PC used for most of this work used 640 x 480 resolution with 256 colours. A higher resolution PC can go up to 1024 x 768 in 256 colours.

The Workstations also have larger monitors than the typical 14 inch PC screen. This means that the image is clearer on the Workstation and that some of the fine detail is lost on the PC screen.

XVision also uses a slightly different colour map than the Workstations so there will be a slight difference in the colours viewed on the PC. XVision cannot deal with the complexity of some of the plots produced and the contoured version of the gaseous detonation, Figure 5a, can't be viewed on the PC.

5.4 AVI Playback Rate

The rate at which the AVI files play on the screen depends on the speed and video card of the computer being used. The slower the machine, the slower the video will run, up to the point where it is moving too slowly to make the video useful. A 486DX-33 with a standard bus was too slow for the gaseous reignition problem, but a 486-66 with a local bus architecture was found to be satisfactory.

5.5 Writing to Video

When writing to standard video, there is a loss of resolution compared to the computer screen. While a PC can display information up to 1280 x 1024, the PAL television standard has a resolution of only 575 x 400. This means that the quality of the picture is not as sharp as that displayed on the PC.

The method of capturing all of the frames of the gmeta file and pasting them into the video editor is very time consuming and repetitious. A simple windows macro was written to automate this process.

The USA and Japan use a different television standard format, NTSC, so this means that videos to be shown in the USA or Japan need to be converted from PAL to NTSC. This can mean a further loss of picture quality, as NTSC has a lower resolution than PAL. The loss of resolution due to the conversion process can be minimised if the original video is made in Super VHS format.

When writing to video using multiple AVI files, it is necessary to pause the video when the current AVI file finishes, and then to restart it when the next AVI file in the sequence is played. This can cause problems if the timing of the pause button is handled incorrectly. If it is pressed too late then the sequence will appear to stop for a moment. If it is pressed too soon, then the last frames will be missed and the video will appear to jump ahead when the next AVI is played.

5.6 Printing

Plots, such as in the gaseous reignition example, can contain a lot of information and hence can take a long time to print. The colour plot of the gaseous detonation using the cell array method is approximately 33 megabytes in Postscript format and takes approximately one and a half hours to print on the WSD Phaser colour printer. When a number of copies are required, or a number of different frames need to be produced, this can mean that the colour printer will be busy for an unacceptably long time. If multiple copies of a single plot are required, a colour photocopier can be used to avoid this delay. This was tried on one of the gaseous detonation contour plots using the CISU colour copier at the Fishermen's Bend site of AMRL. The standard of the photocopies is such that there is no noticeable reduction in picture quality.

6. Conclusions and Recommendations

6.1 Discussion

This report has described the application of the NCAR colour graphics software to the production of both colour contour plots and colour video sequences from the output of selected WSD hydrocodes. The ability to display the output from the hydrocodes using the methods described here offers significant advantages over the previous method of analysing the data; it enables faster and more accurate interpretation of the results, it presents the data in a form which is more easily comprehended by the Sponsor, and others unfamiliar with the subtleties of the work, and provides an enhanced quality of output which is suitable for presentation purposes. In addition, the ability to transfer results to video and view the development of transient phenomena in real time presents a significant aid to the understanding of the basic underlying physical processes involved.

The programs which produce the plots are written in FORTRAN and have so far been applied to computational results from studies on bullet impact and gaseous detonation simulations. The programs have great flexibility however, and will be applied to other Service sponsored tasks within the Division. Two areas of immediate application will be to Navy tasks concerned with modelling the underwater sympathetic detonation of

Composition B, and the performance modelling of the underwater explosive PBXW-115.

The method of transferring the data to video and animating the results is also applicable to other hydrocodes within WSD. The finite element hydrocode DYNA2D is currently used extensively within WSD to model the formation of Explosively Formed Projectiles (EFPs), and the penetration of EFPs within water and other media. The ability to output the data to video to view the deformation of the EFP and surrounding media in real time will be beneficial from both a scientific and presentational viewpoint.

6.2 Equipment Purchases

To provide WSD with a permanent video output capability either a scan converter or video output card has to be purchased. There are several options open to us at this stage: these include buying a video card for the HP Workstation (thereby bypassing the lower resolution PC); buying a scan converter for the PC; buying a video output card for the PC; or buying an Apple Macintosh and video system.

The most direct method would be to transfer the video sequence from the HP Workstation directly onto video. This would save a lot of the time involved with transferring and screen capturing frames and would mean that the video was coming directly from the high resolution HP. However, there are problems with this scenario. As outlined above in Section 5.5, the resolution of the PAL television format is less than the PC anyway - the video format would not be able to take advantage of the higher resolution workstation. The screen refresh rate on the HP Workstation is not fast enough so that the movie display option in the NCAR package could be successfully used to animate the plots. The plots would still have to be manually captured and placed in some form of video editor and made into an animation sequence.

Another option is to purchase an Apple Macintosh computer and accompanying video hardware. The Macintosh is well designed to handle graphics and animation and is able to produce graphics that match the standard of the PC. The Macintosh would need software that would enable the capturing of the frames and their subsequent animation. Provision would have to be made to ensure that the Macintosh was capable of displaying the NCAR gmeta files (which are in a UNIX format) successfully. A potential drawback with the Macintosh would be its lack of compatibility with the IBM PC computers (although there are many programs that allow the reading of PC files).

If purchasing equipment for the PC, there are two main options. The first option involves getting an external converter, which can be connected to any VGA display and converts the output to a Composite video format. The second option is to get a video output card or plug-in video board that is able to output in a Composite video

format. Prices for converters vary from upwards of $1000. When purchasing a video output device there are certain capabilities which should be taken into consideration. One essential requirement is a Flicker-Free Filter to reduce/negate the flickering effect that occurs in converting from PC to video. Without this ability, the quality of the output suffers dramatically. Another necessary function is the ability to zoom in on certain portions of the screen and to ignore the rest of the screen. This allows controlling software such as *Media Player* to be run without appearing on the video.

An example of an external scan converter is the *AVerKey3* that has both the Flicker-Free Filter and zoom abilities. An example of a video board is the *G-Lock VGA+* series. The *CG Suite Pro* in that series allows video input and output, as well as special effects such as video overlay (which allows titling). This is available for approximately $1500.

7. Acknowledgements

The authors would like to thank Dr. Rodney A.J. Borg for providing the simulation data from the bullet impact model, and Mr. Eric Northeast for general computer support and assistance with the colour printer. We thank Mr. Jim Nicholls of the Photography and Applied Imaging Section, AMRL, for the loan of the scan converter, for editing the video tapes, and for providing advice on video output formats. Thanks also to Mr. Peter Collins for providing advice on video editing software and video output cards, and to Mr. Brian Whiffen for the loan of the Super-VHS video recorder.

8. References

1. NCAR Graphics Contouring and Mapping Tutorial, University Corporation for Atmospheric Research, Unix Version 3.2, Document Version 2.0, May 1993

2. NCAR Graphics Fundamentals Manual, University Corporation for Atmospheric Research, Unix Version 3.2, Document Version 1.0, May 1993

3. "Projectile Impact of Australian Composition B", R.J. Swinton and M.C. Chick, DSTO Technical Report, File No. G6 4/8-4574, in publication.

4. "Numerical Simulation of Bullet Impact Experiments using the Forest Fire Reaction Rate Model", R.A.J. Borg and D.A. Jones, DSTO Technical Report, in preparation.

5. "Development of a Two-Dimensional Multi-Material Eulerian Hydrocode to Model Shock Initiation of Ideal and Non-Ideal Condensed Phase Explosives", D.A. Jones, D.L Kennedy and R.A.J. Borg, DSTO Technical Report, in preparation.

6. "Numerical Simulation of Detonation Transfer Between Gaseous Explosive Layers", D.A. Jones, R. Guirguis, and E. Oran, *MRL Research Report* MRL-RR-1-89, 35pp, July 1989.

7. "Detonation Transmission in Layered Explosives", D.A. Jones, M. Sichel, E.S. Oran and R. Guirguis, *Proceedings of The 23rd Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, PA, pp.1805-1811, 1990.

8. "Numerical Simulation of Layered Detonations", D.A. Jones, M. Sichel, R. Guirguis and E.S. Oran, *Progress in Astronautics and Aeronautics*, **133**, 202-220, 1991.

9. "Numerical Simulation of Detonation Transmission", E.S. Oran, D.A. Jones and M. Sichel, *Proceedings of the Royal Society of London, Series A*, **436**, 267-297, 1992.

10. "Ignition in a Complex Mach Structure", E.S. Oran, J.P. Boris, D.A. Jones and M. Sichel, *Progress in Astronautics and Aeronautics*, **153**, 241-252, 1993.

11. "Reignition of Detonation by Reflected Shocks", D.A. Jones, E.S. Oran and M. Sichel, *Shock Waves*, **5**, 47-57, 1995.

12. "The Influence of Cellular Structure on Detonation Transmission", D.A. Jones, G. Kemister, E.S. Oran and M. Sichel, submitted to *Shock Waves*.

9. Appendices

9.1 Bullet Impact Program

```
      PROGRAM BULLET
C       Basis of program that produces colour contours
C       of the bullet impact model.  This code was
C       incorporated into the hydrocode
C
      INTEGER MCX, MBX, MCY, MBY, MCM
C
      PARAMETER (MCX=600,MBX=MCX+1)
      PARAMETER (MCY=100,MBY=MCY+1)
      PARAMETER (MCM=4)
      PARAMETER (FCUT = 1.0E-6)
      parameter(numcont = 15, nogrps = 2, kmap = 1000)
C
      INTEGER NCX, NBX, NCY, NBY
C
C
C-----------------------------------------------------------------------
      REAL RHO(MCX,MCY), VX(MCX,MCY), VY(MCX,MCY)
      REAL E(MCX,MCY), PRS(MCX,MCY)
      REAL RHOP(MCM,MCX,MCY), EP(MCM,MCX,MCY), FRAC(MCM,MCX,MCY)
C
      REAL ISWAP, OMEGA, WA

      real rx, ry, a1, a2, b1, b2

      integer colors(numcont+1)
C Work arrays for area maps
      integer lrwk,liwk,lmap,nwrk
      integer iwrk(mcx*200)
      integer map(mcx*kmap), iarea(nogrps), igrp(nogrps)
      real rwrk(mcx*200), xwrk(mcx*200), ywrk(mcx*200)

C Data array
      real vfhe(mcx,mcy), vfbull(mcx,mcy)
      real omega1(mcx,mcy)
      real telaps

C Barheight and titlesize
      real barhgt,titlesize
```

```
C Contour levels values array
      real contlev(numcont)

      character*20 cycle, ctime
      character*40 header

C Array of contour level values as characters
      character*10 lbls(numcont+1)

      external fill
      external color

C
c     OPEN (UNIT=16,FILE='pjump')
C

C
      CALL INITAL
      DTSAFE = DELTAT


      call opngks
      lrwk = mcx * 200
      liwk = mcx * 200
      lmap = mcx * kmap
      nwrk = mcx * 200

C     * * * Contour Levels  * * *
c     Read contour levels from file
      open(33,file='/users/borgr/bullet/contlevel',status='old')
      read(33,*)(contlev(i),i=1,numcont)
      close(33)

C         Set up color table
      call color

      rx=real(ncx)
      ry=real(ncy)
      a1=0.05
      a2=0.95
      b1=0.5
      b2=ry/rx * (a2-a1) + b1

      call set(a1,a2,b1,b2,1.0,rx,1.0,ry,1)
      call cpsetc('ILT', ' ')
      call cpseti('SET', 0)
```

```
C       Initialize Areas
      call arinam(map, lmap)

C       Turn off automatic contouring and set number of contour levels
      call cpseti('CLS - Contour Level Selection Flag',0)
      call cpseti('NCL - Number of Contour Levels', numcont)

C       Set the contour level values and set contour color to 1 - black
      do 600, i=1, numcont
        call cpseti('PAI - Parameter Array Index', i)
        call cpsetr('CLV - Contour Level Values', contlev(i))
        call cpseti('CLC - Contour Line Color Index',1)
 600    Continue

C       Perform contouring operation
      call cprect(prs, mcx, ncx, ncy, rwrk, lrwk, iwrk, liwk)

C       Add contours to area map
      call cpclam(prs, rwrk, iwrk, map)

C       Set fill style to solid, and fill contours
      call gsfais(1)
      call arscam(map,xwrk,ywrk,nwrk,iarea,igrp,nogrps,fill)

C       Draw Perimeter
      call cpback(prs, rwrk, iwrk)

C       Get screen positions for label bar
      call getset(xmin,xmax,ymin,ymax,dum1,dum2,dum3,dum4,idum)
      barhgt = 5.0
      yrange=ymax-ymin
      ybot = ymin - yrange/ barhgt
      ytop = ymin - yrange/ (4.0*barhgt)

      do 700, i=1,numcont
        colors(i) = i+1
        call cpseti('PAI - parameter array index',i)
        call cpgetr('CLV - contour level values',CLV)
C         Specify the format of the label values
        write (lbls(i),'(E8.2)') CLV

 700    Continue
      colors(numcont+1) = numcont+2
```

```
C       Draw the label bar
        call lblbar (0,xmin,xmax,ybot,ytop,numcont+1,1.0,0.5,
     +  colors, 1, lbls, numcont, 1)

C       Set up the size of the title
        titlesize = 0.015

C       Draw the title
        cycle='Cycle='//cycle
        ctime='Time='//ctime
        header=ctime//cycle
c       call plchhq (.5*rx, 1.1*ry, header, titlesize, 0.,0.)

c       Draw high explosive outline
        call cpcnrc(vfhe,mcx,ncx,ncy,0.0,1.0,0.5,1,-1,0)
c       Draw bullet outline
        call cpcnrc(vfbull,mcx,ncx,ncy,0.0,1.0,0.5,1,-1,0)

        call frame

      ENDIF

  300 CONTINUE
c     CLOSE (UNIT=16)
      call clsgks
C

      STOP
C
      END
```

9.2 Gaseous Detonation Program -Colour Contouring Method

```
      program ArgonContouring

C     Produces colour contours of a gaseous detonation.
C     Because of the size of arrays involved, it will only run
C     on HP2.

      external fill
      external color

      parameter (ncx=640)
      parameter (ncy=640)
      parameter (numcont = 15)
```

```
c
      real prs(ncx,ncy), tmp(ncx,ncy)
      integer map(600000),iwork(200000),lmap,liwork,lwork
      real xwrk(200000),ywrk(200000),rwrk(200000)
      integer iarea(2),igrp(2)
      integer colors(numcont+1)
      integer num_frames

C Contour levels values array
      real contlevel(numcont)

C Barheight and titlesize
      real barhgt,titlesize

C Array of contour level values as characters
      character*10 lbls(numcont+1)

C* * * Contour Levels * * *
  data contlevel / 1.1,
    + 2.0,
    + 3.0,
    + 4.0,
    + 5.0,
    + 8.0,
    + 10.0,
    + 12.0,
    + 14.0,
    + 16.0,
    + 18.0,
    + 20.0,
    + 24.0,
    + 28.0,
    + 32.0 /

c ---------------------------------------------------------------

      lmap = 600000
      liwork = 200000
      lwork = 200000
      nwrk = 200000
      nogrps = 2
      num_frames = 18

c ---------------------------------------------------------------
```

```
      open (unit=12, file='/scratch/users/kemister/argon/argon_cell2.out
     +',status='old')
c
      call opngks

      call color
c
c -------------------------------------------------------------
      do 100 k=1,num_frames
c

C Read in the next set of pressure values
      do 5 j=1,ncy
    5 read (12,190) (prs(i,j),i=1,ncx)
      read(12,*)

C Read in the next set of temperature values
      do 6 j=1,ncy
    6 read (12,190) (tmp(i,j),i=1,ncx)
      read(12,*)

C Adjust the pressure to fit within valid range
      do 50 j=1,ncy
      do 50 i=1,ncx
      if(prs(i,j).lt.1.0e+05) prs(i,j)=1.0e+05
      prs(i,j)=prs(i,j)/1.0e+05
   50 continue

C Adjust the temperature to fit within valid range
      do 60 j=1,ncy
      do 60 i=1,ncx
      if(tmp(i,j).lt.300.) tmp(i,j)=300.0
   60 continue


C     Initialize Areas
            call arinam(map, lmap)
C     Turn off automatic contouring and set number of contour levels
            call cpseti('CLS - Contour Level Selection Flag',0)
            call cpseti('NCL - Number of Contour Levels', numcont)

C     Set the contour level values and set contour color to 1 - black
      do 300, i=1, numcont
            call cpseti('PAI - Parameter Array Index', i)
            call cpsetr('CLV - Contour Level Values', contlevel(i))
            call cpseti('CLC - Contour Line Color Index',1)
```

```
 300  Continue

C     Initialise contour map
            call cprect(prs, ncx,ncx,ncy, rwrk, lwork, iwork, liwork)
C     Add contours to area map
            call cpclam(prs, rwrk, iwork, map)
C     Set fill style to solid, and fill contours
            call gsfais(1)
            call arscam(map, xwrk, ywrk, nwrk, iarea, igrp, nogrps, fill)
C     Draw Perimeter
            call gridal(64,0,64,0,0,0,5)

C     Get screen positions for label bar
            call getset(xmin,xmax,ymin,ymax,dum1,dum2,dum3,dum4,idum)
            barhgt = 7.0
            ybot = ymin / barhgt
            ytop = ymin - ymin/barhgt

            do 400, i=1,numcont
                  colors(i) = i+1
                  call cpseti('PAI - parameter array index',i)
                  call cpgetr('CLV - contour level values',CLV)
C           Specify the format of the label values
                  write (lbls(i),'(F6.1)') CLV

 400        Continue
            colors(numcont+1) = numcont+2
C     Draw the label bar
            call lblbar (0, xmin, xmax, ybot, ytop, numcont+1, 1.0, 0.5,
            +colors, 1, lbls, numcont, 1)

C     Get screen positions for title
            call getset(vpl,vpr,vpb,vpt,wl,wr,wb,wt,ll)
C     Set up the size of the title
            titlesize = 0.4
            titlesize = titlesize * (1.0 - vpt)
            y = 1.0 - .5 * (1.0 - vpt)
            call set (0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1)
C     Draw the title
            call plchhq (.5, y, 'Color Contours', titlesize, 0.,0.)

            call frame
 100  Continue
```

```
 190 format(1x,10e12.5)
c
    call clsgks
c
    close(unit=12)
c
    stop
    end


c -------------------------------------------------------------
     Subroutine Fill (xwrk, ywrk, nwrk, iarea, igrp, nogrps)
C       Defines how the contours are filled in.

C       Working arrays for areas routine
        dimension xwrk(*),ywrk(*),iarea(*),igrp(*)

        do 100 I=1,nogrps
        if (igrp(i).eq.3) iarea3=iarea(i)
C       If the group identifier matches contour then give it
C       an area identifier. nogrps is the number of different groups
 100    Continue

        if (iarea3. gt. 0) then
C       If the area is defined by 3 or more points, fill it
C       Sets the color index for each of the areas.
C       +1 is added to index as colors for contours start at number 2
                call gsfaci(iarea3+1)
                call gfa(nwrk,xwrk,ywrk)
        end if

        return
        end
c -------------------------------------------------------------



c -------------------------------------------------------------
        Subroutine color
C Define color indices

C       BACKGROUND COLOR
C White
        CALL GSCR(1,  0, 1.0, 1.0, 1.0)

C       FOREGROUND COLOR  (labels, perimeter, etc.)
```

```
C Black
      ALL GSCR(1,  1, 0.0, 0.0, 0.0)

C     CONTOUR COLORS
C White
      CALL GSCR(1,  2, 1.0, 1.0, 1.0)
C Light Blue
      CALL GSCR(1,  3, 0.5, 0.8, 1.0)
C Blue
      CALL GSCR(1,  4, 0.1, 0.2, 1.0)
C Deep blue
      CALL GSCR(1,  5, 0.0, 0.3, 0.8)
C Turquoise
      CALL GSCR(1,  6, 0.0, 0.5, 0.7)
C BlueGreen
      CALL GSCR(1,  7, 0.0, 0.6, 0.6)
C GreenBlue
      CALL GSCR(1,  8, 0.1, 0.85, 0.7)
C Green
      CALL GSCR(1,  9, 0.0, 1.0, 0.0)
C Pale Green
      CALL GSCR(1, 10, 0.5, 1.0, 0.5)
C Yellow Green
      CALL GSCR(1, 11, 0.8, 1.0, 0.2)
C Yellow
      CALL GSCR(1, 12, 1.0, 1.0, 0.0)
C Pale Orange
      CALL GSCR(1, 13, 1.0, 0.8, 0.3)
C Light Orange
      CALL GSCR(1, 14, 1.0, 0.6, 0.0)
C Orange
      CALL GSCR(1, 15, 1.0, 0.45, 0.0)
C Dark Orange
      CALL GSCR(1, 16, 1.0, 0.25, 0.0)
C Red
      CALL GSCR(1, 17, 1.0, 0.0, 0.0)

      return
      end
c -------------------------------------------------------------
```

9.3 Gaseous Detonation Program - Cell Array Method

```
      Program ColorCellsPlot

C     Produces colour plot of gaseous detonation using cell array
C     method

      parameter (ncx=640)
      parameter (ncy=640)
      external color


c -----------------------------------------------------------------
      open (unit=12, file='/scratch1/users/kemister/argon/argon_cell2.ou
     +t',status='old')

      call opngks

      call genplot(12,ncx,ncy,mcx,mcy,color)
      close(unit=12)

      call clsgks

      stop
      end
c -----------------------------------------------------------------


c -----------------------------------------------------------------
      Subroutine genplot(unit,ncx,ncy,mcx,mcy,color)
      parameter(numcont = 15, nogrps = 2, workmap = 2000,colnum=161)
      integer unit,ncx,mcx,mcy

C     Work arrays for area maps
            integer lrwk,liwk,lmap,nwrk
            integer iwrk(ncx*200)
            integer map(ncx*workmap)
            real rwrk(ncx*200), xwrk(ncx*200), ywrk(ncx*200)
C     Data array
            real prs(ncx,ncy)
            real tmp(ncx,ncy)
C     Contour levels values array
            real range(colnum)
            real pmax,pmin,pgood,lownum
            integer colia(ncx,ncy)
            integer num_frames
```

```
C       Array of contour level values as characters

        external color

C       Regions of interest.
C       Colors are produced between pmin and pmax.
C       Lownum is the number of colours between pmin and pgood.
C       Colnum-lownum is the amount of remaining colours that
C       represent the range between pgood and pmax.
              lownum = 140.0
              pmin = 1.0
              pgood = 15.0
              pmax = 30.0


              lrwk = ncx * 200
              liwk = ncx * 200
              lmap = ncx * workmap
              nwrk = ncx * 200

              num_frames = 18

C       Set up color table
        call color

 750    format ("Running.  Please wait...")
        print 750

 190    format(1x,10e12.5)

C       Calculate which colours correspond to which regions
              range(1) = pmin
              do 220 i=2,lownum+1
              range(i) = range(i-1) + ((pgood-pmin)/lownum)
 220    Continue

        do 230 i=lownum+2,colnum
              range(i) = range(i-1) + ((pmax-pgood)/(colnum-lownum))
 230    Continue


C       Keep generating frames until the end of the data is reached
              do 100 k=1,num_frames
```

```
C       Read in the next set of pressure values
              do 5 j=1,ncy
 5                  read (12,190) (prs(i,j),i=1,ncx)
              read(12,*)

C       Read in the next set of contour values
              do 6 j=1,ncy
 6                  read (12,190) (tmp(i,j),i=1,ncx)
              read(12,*)

C       Adjust the pressure values so that they are within
C       a valid range.
              do 50 j=1,ncy
              do 50 i=1,ncx
                   if (prs(i,j).lt.1.0e+05) prs(i,j)=1.0e+5
                   prs(i,j)=prs(i,j)/1.0e+05
 50           Continue

C       Adjust the temperature values so that they are within
C       a valid range.
              do 60 j=1,ncy
              do 60 i=1,ncx
                   if(tmp(i,j).lt.300) tmp(i,j) = 300.0
 60           Continue


C       Put color indices into the cell array according to
C       which range the data falls into.
        do 250 i=1,ncx
              do 280 j=1,ncy
                   col = 0

 290               if (col.lt.colnum) then
                        col = col + 1
                        if(prs(i,j).lt.range(col)) then
                             colia(i,j) = col
                        else
                             goto 290
                             end if
                        end if
 280           Continue
 250   Continue

C       Display the cell array
              call gca(0.0, 0.0, 1.0, 1.0, ncx,ncy,1,1,ncx,ncy,colia)
```

```
C       Draw a perimeter
              call gridal(0,199,0,199,0,0,5,1,1)

              call frame
 100    Continue

        return
        end
c -------------------------------------------------------------


c -------------------------------------------------------------
        Subroutine color
C       Define color indices

              integer index
              real red,green,blue
              real coldiff
              integer colstep


C       BACKGROUND COLOR
C  White
        CALL GSCR(1,  0, 1.0, 1.0, 1.0)

C       FOREGROUND COLOR  (labels, perimeter, etc.)
C  Black
        CALL GSCR(1,  1, 0.0, 0.0, 0.0)

C       CONTOUR COLORS
C  White
        CALL GSCR(1,  2, 1.0, 1.0, 1.0)

        colstep = 40

        index = 1
        red = 1.0
        green = 1.0
        blue = 1.0
        coldiff = 1. / colstep


C       Define colours White to Blue
              do 100 i=1,colstep
                    call gscr(1,index+i,red,green,blue)
                    red = red - coldiff
```

```
                        green = green - coldiff
  100   Continue


        index = colstep + 1
        red = 0.0
        green = 0.0
        blue = 1.0
C       Blue to Green
              do 200 i=1,colstep
                    call gscr(1,index+i,red,green,blue)
                    blue = blue - coldiff
                    green = green + coldiff
  200   Continue


        index = (2*colstep) + 1
        red = 0.0
        green = 1.0
        blue = 0.0
C       Green to Yellow
              do 300 i=1,colstep
                    call gscr(1,index+i,red,green,blue)
                    red = red + coldiff
  300   Continue


        index = (3*colstep) + 1
        red = 1.0
        green = 1.0
        blue = 0.0
C       Yellow to Red
              do 400 i=1,colstep
                    call gscr(1,index+i,red,green,blue)
                    green = green - coldiff
  400   Continue

        return
        end
c -------------------------------------------------------------
```

9.4 Triple Point Trajectories Program

```
      Program TriplePointMarking

      parameter (ncx=960,ncy=320)
      external color


c -----------------------------------------------------------------
      open (unit=12, file='/scratch1/users/kemister/argon/argon_cell1.ou
     +t',status='old')

      call opngks

      call genplot(12,ncx,ncy,mcx,mcy,color)
      close(unit=12)

      call clsgks

      stop
      end
c -----------------------------------------------------------------


c -----------------------------------------------------------------
      Subroutine genplot(unit,ncx,ncy,mcx,mcy,color)
      parameter(numcont = 15, nogrps = 2, workmap = 2000)
      integer unit,ncx,mcx,mcy

C     Data array
            real prs(ncx,ncy)
            real tmp(ncx,ncy)
C     Contour levels values array
            real tripmark
            integer colia(ncx,ncy)

      external color
C     Value of pressure to mark above as triple point
            tripmark = 30.0

C     Set up color table
            call color

   10 format ("Running.  Please wait...")
      print 10
```

```
 190   format(1x,10e12.5)


C      Initialise color array to 0 (white screen)
            do 50 i=1,ncx
                  do 60 j=1,ncy
                        colia(i,j) = 0
 60           Continue
 50    Continue


C      Read in all the data and set up color cell index
            do 100 k=1,13
                  do 110 j=1,ncy
 110                  read (12,190) (prs(i,j),i=1,ncx)
            read(12,*)

            do 120 j=1,ncy
 120                  read (12,190) (tmp(i,j),i=1,ncx)
            read(12,*)

            do 150 j=1,ncy
            do 150 i=1,ncx
                  if (prs(i,j).lt.1.0e+05) prs(i,j)=1.0e+5
                  prs(i,j)=prs(i,j)/1.0e+05
 150        Continue

            do 160 j=1,ncy
            do 160 i=1,ncx
                  if(tmp(i,j).lt.300) tmp(i,j) = 300.0
 160   Continue

C      For each value in the run if it is above set pressure then mark it
            do 200 i=1,ncx
                  do 210 j=1,ncy
                        if(prs(i,j).gt.tripmark) then
                              colia(i,j) = 1
                        end if
 210                  Continue
 200        Continue


 100   Continue

C      Call to set up color cell array
            call gca(0.0, 0.0, 1.0, 1.0, 960,320,1,1,960,320,colia)
```

```
            call gridal(0,199,0,199,0,0,5,1,1)
            call frame

        return
        end
c ---------------------------------------------------------------

c ---------------------------------------------------------------
        Subroutine color
C       Define color indices
C       BACKGROUND COLOR
C   White
        CALL GSCR(1,  0, 1.0, 1.0, 1.0)

C     FOREGROUND COLOR  (for points)
C   Black
        CALL GSCR(1,  1, 0.0, 0.0, 0.0)

        return
        end
c ---------------------------------------------------------------
```

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PAGE CLASSIFICATION UNCLASSIFIED 2. PRIVACY MARKING/CAVEAT (OF DOCUMEN

3. TITLE	4. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Colour graphics for hydrocode simulations	Document (U) Title (U) Abstract (U)

5. AUTHOR(S)	6. CORPORATE AUTHOR
A. Doyle, D.A. Jones and G. Kemister	Aeronautical and Maritime Research Laboratory PO Box 4331 Melbourne Vic 3001

7a. DSTO NUMBER	7b. AR NUMBER	7c. TYPE OF REPORT	8. DOCUMENT DATE
DSTO-GD-0059	AR-009-358	General Document	December 1995

9. FILE NUMBER	10. TASK NUMBER	11. TASK SPONSOR	12. NO. OF PAGES	13. NO. OF REFERENC
510/207/0368	DST 93/101	DSTO	57	12

14. DOWNGRADING/DELIMITING INSTRUCTIONS	15. RELEASE AUTHORITY
	Chief, Weapons Systems Division

16. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DI OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

17. DELIBERATE ANNOUNCEMENT

Announcement of this report is unlimited

18. CASUAL ANNOUNCEMENT YES/NO (Cross out whichever is not applicable)

19. DEFTEST DESCRIPTORS

Simulation; Detonation; Composition B explosives; Underwater explosions; Hydrocodes

20. ABSTRACT

This report describes the production of colour graphical output for the two-dimensional finite difference reactive hydrocodes currently used by Weapons Systems Division. The programs use the NCAR graphics package and show how the careful choic appropriate colour schemes can significantly enhance the usefulness of the output from the finite difference codes. Application recent work in WSD are described, including simulation of the initiation of detonation in Composition B by bullet impact, and reignition of gaseous detonation following an abrupt area change. The report also discusses several ways in which a sequence c colour plots can be animated and then transferred to videotape, and a videotape sequence of the time dependent cellular struct of a gaseous detonation front was made to illustrate the implementation of the recommended procedure. Now that the capabili of the software have been fully demonstrated these colour graphical and video techniques will be used on Service Sponsored Tasks to more effectively communicate the results and outcomes to the sponsors.

Colour Graphics for Hydrocode Simulations

A. Doyle, D.A. Jones and G. Kemister

(DSTO-GD-0059)

DISTRIBUTION LIST

DEFENCE ORGANISATION

Defence Science and Technology Organisation

Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
} shared copy
Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington (Doc Data Sheet only)
Scientific Adviser to Thailand MRD (Doc Data Sheet Only)
Scientific Adviser to the DRC (Kuala Lumpur) (Doc Data Sheet Only)
Senior Defence Scientific Adviser/Scientific Adviser Policy and Command (shared copy)
Navy Scientific Adviser (3 copies Doc Data Sheet)
Scientific Adviser - Army (Doc Data Sheet only)
Air Force Scientific Adviser
Director Trials

Aeronautical and Maritime Research Laboratory

Director
Chief, Weapons Systems Division
Mr M.C. Chick
Mr A. Doyle
Dr D.A. Jones
Dr G. Kemister
Mr E. Northeast
Dr R.A.J. Borg
Mr P. Elischer, SSMD

DSTO Library

Library Fishermens Bend
Library Maribyrnong
Main Library DSTOS (2 copies)
Library, MOD, Pyrmont (Doc Data Sheet only)

Defence Central

OIC TRS, Defence Central Library
Officer in Charge, Document Exchange Centre, 1 copy
DEC requires, as permitted by the release limitations, the following copies to meet exchange agreements under their management:
*US Defence Technical Information Centre, 2 copies
*UK Defence Research Information Centre, 2 copies
*Canada Defence Scientific Information Service, 1 copy
*NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy
The Director, Qinghua University, Beijing, 1 copy
The Librarian, Beijing University of Aeronautics, 1 copy
The Librarian, Institute of Mechanics, Chinese Academy of Sciences, 1 copy

DISTRIBUTION LIST
(Continued)

(DSTO-GD-0059)

Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet only)

Army

Director General Force Development (Land) (Doc Data Sheet only)
ABCA Office, G-1-34, Russell Offices, Canberra (4 copies)
NAPOC QWG Engineer NBCD c/- DENGRS-A, HQ Engineer Centre

Navy

ASSTASS, APW2-1-OA2, Anzac Park West, Canberra (Doc Data Sheet only)

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Senior Librarian, Hargrave Library, Monash University

OTHER ORGANISATIONS

NASA (Canberra)
AGPS

ABSTRACTING AND INFORMATION ORGANISATIONS

INSPEC: Acquisitions Section Institution of Electrical Engineers
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
American Society for Metals
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US